AUSTRALIAN OS9 NEWSLETTER

EDITOR :
Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

(07) 345-5141

JULY 1989

ATTR Syntax: Attr filename [permissions] Usage : Examine or change the security permissions of a file Opts: -perm = turn off specified permission perm= turn on specified permission -a = inhibit rms : d - directory file s - nc to owner w - write permit to own r - read permit to public pw - te permit to public BACKUP Syntax ge : Copies all data from one de ead error occurs s = single writes BASIC09 Syntax : Basic0 ge BUILD Syntax: Build filenar s from standard input CHD S nge working directory to specifi > Usage: Change execution directory to specified path CMP Syntax: Cmp filename1 filename2 Usage : File comparison utility COBBLER Syntax: Cobbler devname Usage : Creates OS-9 bootstrap file from current boot CONFIG Syntax: Config Usage : Create system boots and system disks COPY Syntax data from one fil E Syntax : Date [t Opts: t = specify ame> Usage : Check directory for wor sters -m = save of unused cluster nly -o = print <devname> {<devn : Del [-x] filenam s : -x = delete x: Deldir directo yntax: Dir [e] [x] the file names y x=print executi] Usage : Display s converted characters to standard output DSAVE Syntax : Dsave [-opts] [dev] [pathname] Usage : Generates procedure file to copy all files in a directory system Opts : -b make a system disk by using OS9boot if present -b=<path> = make system disk using path as source i = indent for directory levels l = do not process b makdir command o num K ECHO Syn tandard output ED oriented text edito s text error messages for given error numbers EX Syntax: ex <modname> Usage: Chain to the given module FORMAT Syntax : Format <devname> Usage : Initializes an OS-9 diskette Opts ; R - Ready L - Logical format only "disk name" 1/2 number of sides 'No of

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group

EDITOR : Gordon Bentzen

HELPERS : Bob Devries and Don Berrie

SUPPORT : Brisbane OS9 Level 2 User Group.

Welcome to yet another edition of our newsletter. Boy! These months certainly seem to roll around quickly, it must be that the months are getting shorter, it seems that we no sooner complete one newsletter and post it off than we are short of time for the next - again!

Well after last month's editorial I guess that you do not want any more pleas for material, so I will just get on with the good news.

I am pleased to report that subscription renewals for another twelve months have already begun, all be it slowly at this stage. As stated in our July 1988 edition of this newsletter, we will require at least a membership of twenty to justify the effort that goes into production, so please help us by sending your renewal form and subscription by the end of August.

Apology:- The editorial of the June edition included reference to some new public domain software from the U.S.A. which was included in that newsletter. Did you have trouble finding it? Well we slipped up last month and did not include that material. Sorry. Bob has included his comments on that software in this edition.

Don has also included the first part of a Tutorial on EDIT, the OS9 Macro text editor, which will no doubt assist many readers.

If you have been following the Data Base series by Bob Devries (and I hope that you have) you will find some more of the 'C' source code by Bob in these pages.

Phil Frost, a user group member from Kalgoorlie, has been good enough to send a Basic09 procedure to share with us. This is a simple programme which demonstrates Ohms law. The source code is included in this edition. Many thanks Phil for sharing it with us.

As a final comment this month I must say that we expected last month's editorial to prompt some comments from the membership. Very little has been received, in fact no comment would be the most accurate description.

We are keen to present something that is interesting and hopefully useful, so please drop us a short note and tell us what you like and also what you don't like. Perhaps you could make a note or two on your subscription renewal form so that we may be able to include something to suit your special interests.

COCO-LINK Magazine: Just a reminder for those interested in a wider section of CoCo systems. CoCo-Link is produced in South Australia and is well worth your support. Contact the editor, Robbie Dalzell on (08) 386 1647 or write;

CoCo-Link 31 Nedland Cres. PT.NOARLUNGA STH S.A. 5167

The bi-monthy magazine will cost you $8.00 joining fee plus $12.00 a year subscription.

Until next month, happy computing.
Regards, Gordon Bentzen.

Phil Frost
25 Cheetham Street
KALGOORLIE 6430 W.A.

10th June 1989

Dear Gordon & Crew

Well I have finally written a programmme, this is my first effort, I hope you like it. It is an educational programme on OHM'S LAW and can be used to work out values in electronic circuits. The programme is written in Basic09, as an executable module in the CMDS directory, OHMS. The source code is the root directory, OHMS, and as a list file Ohms_Law.

So cheers for now.

Phil.

```
PROCEDURE ohms
(* ohms law
(* Phil Frost  20th April 1989
(* a program to demonstrate OHM'S LAW
(*
DIM ohms,volts,amps:REAL
DIM select:STRING[1]
DIM choice:BYTE
(*
ohms:=0
volts:=0
amps:=0
(*
10 PRINT CHR$(12)
PRINT
PRINT TAB(29); "***** OHM'S LAW *****"
PRINT
PRINT "          When a voltage is applied to a resistive circuit the current in"
PRINT "      amperes will be proportional to the voltage, and inversly proportional"
PRINT "      to the resistance in ohms"
PRINT
PRINT TAB(16); "This relationship is represented by the formulas"
PRINT
PRINT TAB(14); "E = I x R     or     I = E / R     or     R = E / I"
PRINT
PRINT TAB(20); "Where E = Volts    I = Amps    R = Ohm's"
PRINT
PRINT TAB(21); "<1> To find the Current in a circuit "
PRINT
PRINT TAB(21); "<2> To find the Resistance of a circuit "
PRINT
PRINT TAB(21); "<3> To find the Voltage in a circuit "
PRINT
PRINT TAB(21); "<4> Quit "
PRINT  \ PRINT
PRINT TAB(21); " Select: ";  \ INPUT choice
ON choice GOTO 100,200,300,400
100 PRINT CHR$(12);
ohms:=0
volts:=0
amps:=0
PRINT  \ PRINT  \ PRINT
```

```
PRINT "   To find the Current (amps) in a circuit, where the Ohms and Volts are known"
PRINT  \ PRINT
PRINT TAB(35); "E = I / R "
PRINT  \ PRINT
PRINT TAB(20); "Where E = Volts    I = Amps    R = Ohm's"
PRINT  \ PRINT
INPUT "          Enter the resistance in Ohms :",ohms
PRINT  \ PRINT
INPUT "          Enter the Voltage           :",volts
PRINT  \ PRINT
amps=volts/ohms
PRINT "          The Current in the circuit is";
PRINT USING "r8.2",amps;
PRINT " Amps : "
PRINT  \ PRINT
PRINT "          <A>nother or <M>enu  ";
INPUT select
IF select="a" OR select="A" THEN
GOTO 100
ELSE
IF select="m" OR select="M" THEN
ENDIF
ENDIF
GOTO 10
200 PRINT CHR$(12)
ohms:=0
volts:=0
amps:=0
PRINT  \ PRINT  \ PRINT
PRINT "   To find the resistance in a circuit, where the Voltage and Amperage are known "
PRINT  \ PRINT
PRINT TAB(35); "R = E / I "
PRINT  \ PRINT
PRINT TAB(20); "Where E = Volts    I = Amps    R = Ohms"
PRINT  \ PRINT
INPUT "          Enter the Current in Amps  :",amps
PRINT  \ PRINT
INPUT "          Enter the Voltage          : ",volts
PRINT  \ PRINT
ohms=volts/amps
PRINT "          The Resistance in the circuit is : ";
PRINT USING "r8.2",ohms;
PRINT " Ohms "
PRINT  \ PRINT
PRINT "          <A>nother or <M>enu  ", \ INPUT select
IF select="a" OR select="A" THEN
GOTO 200
ELSE
IF select="m" OR select="M" THEN
ENDIF
ENDIF
GOTO 10
ohms:=0
volts:=0
amps:=0
300 PRINT CHR$(12)
PRINT  \ PRINT  \ PRINT
```

```
PRINT "  To find Voltage in a circuit, where the Resistance and Current are known"
PRINT  \ PRINT
PRINT TAB(35); "E = I x R "
PRINT  \ PRINT
PRINT TAB(20); "Where E = Volts    I = Amps    R = Ohms"
PRINT  \ PRINT
INPUT "           Enter the resistance in Ohms : ",ohms
PRINT  \ PRINT
INPUT "           Enter the current in Amps    : ",amps
PRINT  \ PRINT
volts=amps*ohms
PRINT "           The Voltage in the circuit is : ";
PRINT USING "r8.2",volts;
PRINT " Volts "
PRINT  \ PRINT
PRINT "           <A>nother or <M>enu  ", \ INPUT select
IF select="a" OR select="A" THEN
GOTO 300
ELSE
IF select="m" OR select="M" THEN
ENDIF
ENDIF
GOTO 10
400 END
```

oooooooooo0000000000ooooooooooo

EDIT : A tutorial on the OS-9 Macro Text Editor

The OS-9 editor is a powerful but easy to use text editor. Some of it's features are small size, multiple buffers, and edit macros. The editor occupies only 5K of memory, leaving plenty of space for large text files. The editor requires at least 2K of free RAM to run. The editing of more than one file at the same time can also be undertaken since it allows the creation of 2 or more buffers. And you can create macros that let you customise the editor for personal use. All OS-9 shell commands are usable inside the text editor, and files larger than the buffer can be successfully edited.

After using EDIT you will find that there are a few commands that you will use more frequently than others. Here is a list of the ones that the majority of users find most useful.

^ [or <CTRL>+7]	move to start of text
/	move to end of text
+n	move forward n lines
-n	move backward n lines
Cn S1 S2	replace S1 with S2 n times
Dn	delete n lines
En S1	extend n lines with S1
Sn S1	search for n occurrences of S1
Ln	list n lines
Q	quit

This is a very small list of the possible commands that can be used with EDIT. In general, n is a number; leaving it out of the command implies 1. If an "*" is used instead of n, then the number is the largest possible integer (65535 in an 8 bit system). S1 and S2 are any string that you wish to use. They are always delimited with some character. It doesn't matter what the character is, as long as it is used throughout the command line. When any of the above commands are used, the edit pointer is left at the beginning of the target line. The exception is "/", which leaves the pointer after the last line.

To start the editor, type :

```
EDIT filename1 [filename2 #nK]
```

If filename1 exists, a temporary file named SCRATCH (yes, it is in upper case) will be created. On termination of an edit session, the original will be deleted, and SCRATCH will be renamed. You must have DEL and RENAME in memory or your execution directory. In order to successfully edit a file, the file read and write attributes need to be set. If filename1 does not exist, it will be created. If filename2 is specified, the original file is not modified, and the newly edited file will be named filename2. The "#nK" is the standard OS-9 memory allocation modifier. If no memory size is specified, then 8K (the system default) will be used. The editor prints a prompt in much the same manner as OS-9. It prompts with:

```
E:
```

The editor assumes that any character in position 1 of a line signifies the start of a command string. A space in position 1 tells the editor that what follows is to be inserted as a new line. If the editor cannot interpret a command, or the command is not legitimate, the editor responds with:

```
WHAT?
```

One of the most common causes of this response stems from neglecting to start a new input line with a space. I normally use only a few commands to move through the text. Armed with +n and -n, I can get anywhere. The ^ and / are also convenient to move to the beginning and the end of the text buffer. These can be duplicated by +* and -*, however,in a large textfile, these alternatives are quite a bit slower. The +* and -* appear to move through all of the textlines, while the ^ and / go directly to their respective destinations. Nevertheless, the effect is the same. When using any of the movement commands, the imaginary edit pointer is moved to some particular line. If you've used only line number oriented editing this may take a little getting used to. Most people find it easy to use after some practice. To search for a particular text string, use S. That is, to search for the string "hello there!", enter:

```
E:S/hello there!/
```

The edit pointer will stop at the beginning of the line in which the first match is found. Remember, the match must be exact. The editor will distinguish between upper and lowercase. Placing a number after the S will cause it to search for that number of occurrences of the target text.

```
E:S5;hello there!;
```

will pass over the target string 4 times, displaying the line each time, and stop at the beginning of the fifth line containing the target string. Notice the differing delimiters used to mark the string in each example. The only requirement is that delimiters are the same, on each individual command line. The command:

```
E:Sstrings
```

would search for the target string "tring", as the delimiter in this instance is s. The C command is used to alter the target string to a specified replacement string. For example, to change the occurrence of "the" to "this" for the next three instances, we could enter:

```
E:C3!the!this!
```

Here the ! delimits the strings. Any other character could be used, so long as we are consistent. A variation which can be used to delete occurrences of words is:

```
E:C3.yes..
```

This would remove the next three occurrences of yes.

The reverse can also be used:

```
E:C::10:
```

would insert 10 at the beginning of the line. Above, when we discussed the most used commands, we mentioned the E command. This command extends the current line with the delimited text. It is especially handy when commenting programme listings, and with a little experience, becomes an extremely fast way of commenting your code. The L command is used to list parts of your text.

```
E:L12
```

will list the next 12 lines of text starting at the current edit pointer location. The edit pointer does not change position. Using L without a number suffix will list the current line, and this is an easy way to determine on which line the edit pointer is situated. The last of our most used commands is the D command. D will delete lines starting from the position of the edit pointer. To remove 3 lines use:

```
E:D3
```

To delete text from the current pointer position to the end of the buffer, use the * modifier. Commands can be chained together. As long as the start in the first column, they can be entered on 1 line. The following command, for example:

```
E:^+3C2/Yes/No/
```

moves the pointer to the top of the buffer, moves down 3 lines, and then changes the next 2 occurrences of Yes to No. You should perhaps think twice about using the following:

```
E:^D*
```

When finishing an editing session, enter Q. The buffer will be written out to the file. Any existing file will be deleted if necessary, and the SCRATCH file will be renamed to the oldname. If things should go awry, you could find that you are left with a file named SCRATCH, and you will have to do some deleting and renaming of your own. In this article, all commands have been presented in upper case to make them more easily readable. This is not necessary, and could just as well have been in lower case. These simpler commands pertaining to the editor are very similar to those of the BASIC09 editor. This is only a small smattering of what can be done with the OS-9 editor, and many things have not been covered. For example, there are 34 commands and 26 pseudo macros in the macro commands section of the editor. It is possible, of course, to create you own macros, and we hope to be able to present an article on this in the near future.

oooooooooo0000000000oooooooooo

The Public Domain Library.
A Review by Bob Devries.

The National OS9 Usergroup public domain library has recently been augmented by the addition of some programmes and other files from the Compu-Serve OS9 Forum in the United States. They came to us by courtesy of Mike Harris of Sydney.

In this review I will briefly cover the contents of the 34 archived files that Mike sent me (Thanks Mike !!!). I have unarchived all the files and they now take up one full 80 track double sided disk and about a third of another. All of the files were archived using the programme 'AR09' which is also available in the public domain library. Quite a lot of the files are actually patch files, including some for use with MODPATCH and some others for use with a programme called IPATCH. This is a specially written patching programme which includes the facility to insert code into a file, which is something MODPATCH cannot do.

Now I will cover each archive file in turn and discuss its contents.

ALIB

This is a library of routines for machine code programmers (and probably C also) which has been written in the format necessary for RMA to use. It comes complete with document files. Written by Bob van der Poel.

BUGS

This is a text file which discusses several bugs which have been found in the OS9 level II system files, and details how to patch them. It was written by Kevin Darling, and was the subject of one chapter in his book Inside OS9 level II.

CC

This is the C source code for a new 'front end' for the Microware C Compiler. It replaces CC1. It is complete with the source code (in C of course) and documentation. Written by Bill Dickhaus.

CC3DISK

This is a patch file for the CC3Disk driver programme to allow the user to read foreign format disks. Author unknown. Document files included.

CHOOSE

Here's a little beauty which a lot of people will probably find useful. Amongst other things, it has the ability to display all the files in the current directory and allows you to choose them one at a time to pass their names to a particular command. It is mouse driven, and written in C. Source and docs provided. Written by Stephen Clark.

CLIBS

These files are a new C standard library written by Karl Kreider to replace the standard library (Clib.l) on the Microware C Compiler. Two versions are provided, one includes the transcendental maths functions. The documents are extensive and written by Mark Griffith (SLED author).

CPATCH

A patch file to allow the C Compiler to use the RMA assembler and the RLINK linker from the Development System. Author unknown.

DATAMO

Datamod by Ron Lammardo is a programme to convert shellscripts to files which are able to be loaded into memory. Includes doc files.

EDCON

An Icon Editor by Toby Farley. Uses the Multi-Vue environment with pull-down menus etc. Includes doc files and aif file and icon.

FLINK

File Link allows a file to have a directory appearance in more than one directory. By K.R.McMurdo. Includes doc file.

FORKSH

Forks a shell if a key is pressed within a user selectable time limit. Useful for in startup files. By Pete Lyall. Includes manual and help file. Written in C, and includes source.

GFRMAT

Graphic Format is a front-end programme for the normal FORMAT to pretty it up for use with Multi-Vue. By Mike Haaland. Includes docs, and source in C.

GSHEL2

Version 1.24a of Kent Meyers' patch for the Multi-Vue GSHELL programme. Good increase in speed, and includes some nice extras. This archive is mostly files with a .ipc extension which means that they are for use with IPATCH. Doc file included.

GSHELL

Version 1.24 of Kent Meyers' patch for Multi-Vue GSHELL programme. You'll need this one before the version above. Several other programmes are included in this archive including a directory sort for Multi-Vue.

GSHPAT

A patch for a patch. This one patches the patched GSHELL and allows Multi-Vue to start in 80 column mode. Also supplied with an alternate font set. Docs included. This one is by Mike Haaland.

HDMAKE

This one is a utility for assembly language programmers for easing the use of RMA and RLINK. By Ron Lammardo. Includes assembler source and short doc file.

HEADER

This archive contains 25 C header files a lot of which are not included with the C Compiler. Author unknown. Files are all in C source of course.

HELV

This archive contains a 'Helvitica' style text font, and includes an icon and aif file to start a shell in Multi-Vue with the new font. Small doc file included. Author unknown.

ICONS

Here are 16 different icons for various different programmes (some of which I've never heard of !) with their respective aif files. Great for ideas on how to set up your own applications under Multi-Vue. Author unknown, but a small readme file included.

IPATCH

Here's the archive you'll need to do some of the fancier patches including Kent Meyers' GSHELL patch. This one is quite complex, but includes full documentation. It also includes a patch file for converting PRINTERR from the level one OS9 to work with level II. Written by R.M.Santy.

MV2PAT

Various IPATCH files for system modifications including WindInt, CC3IO and GPort. By Kent Meyers, includes documentation.

MVDEMO

Demonstration programme for use with Multi-Vue including icons etc. Includes C source and document files. By Toby Farley.

MVSKEL

A skeleton programme for C programmers to use for applications programmes for Multi-Vue. Source and docs provided and written by Mark Griffith.

MVTEST

A Basic09 programme written by Kevin Darling for testing out the functions of Multi-Vue. Also useful to see how it's done. No documentation, but source code is commented.

09GIF

Here is a graphics utility for OS9. It allows you to transfer pictures encoded in GIF file format and display them on the colour computer screen. So those people with ColorMax can transfer pictures from RSDOS to OS9. I have been able to transfer pictures from my Commodore Amiga to OS9 and display them. Documents and icon/aif files included. By Chris Babcock.

PLAY

A programme for playing on the CoCo's speaker, files from the Apple MacIntosh or Amiga which were digitized by those computers. Includes a sample file and sparse documentation. Possibly written by Kevin Darling, but I'm not sure.

RSDOS

For those of you who use both RSDOS and OS9, and use SDisk3, here's a utility that will allow you to transfer files straight to and from an RSDOS disk. It even works on 35 track RSDOS disks in an 80 track OS9 drive! Author unknown, but this programme comes included in the package with the CRC-Disto Super Controller II.

SCFED2

Fixes and mods to the SCF manager programme under level II OS9 to allow line recalling AND editing. Includes documents, and is written by Kevin Darling.

SCRIPT

Details and examples on how to set up script files for use with Shellplus. Great stuff ! By Steve Clark.

SHELL21

Version 2.1 of Shellplus including all the modpatch files for enabling some of the options. Includes complete (long) documentation. Written by Ron Lammardo.

SSCRIP

More samples of how to use script files with Shellplus by Steve Clark.

TFLICONS

A few more (9) icons and their aif files for using various programmes under Multi-Vue. By Brian Stretch. Readme file included.

VDD

Modifications to the ramdisk supplied with the Development System. Programme by Volney La Rowe. Readme file included. IPATCH needed for this one.

WINVDG

Various assembler files and mods for window device descriptors using the VDG (32X16) screen with WindInt or GrfInt. Docs included. Written by Bruce Isted.

Well, that's all for now, if you feel you absolutely must have some of these archives, they are available in the usual way from the National OS9 User Group. As I said, this lot altogether takes up two 80 track disks, and of course a lot more in the less dense formats.

Regards,
Bob Devries.

oooooooooo0000000000oooooooooo

The OS9 Operating System

It seems that some users of Microware OS9 Level 1 on a Tandy CoCo system are still using single sided 35 track disk format. If you have disk drives capable of 40 track single sided or 40 track double sided you can configure OS9 to fully utilise such capability. The "standard" versions of Microware Level 1 OS9 as distributed by Tandy for the CoCo did not provide for such drives, however there are various means of modifying the operating system to achieve the desired result.

The main problem is that the disk driver, CCDisk, for the CoCo is hard coded for 35 track single sided drives. Further, this driver does not look at the device descriptors D0, D1, D2, D3 for track count, number of sides or head step rate. So this means that both the device driver, CCDisk and also the descriptors D0, D1 etc., need to be patched.

Now, as I have commented in earlier articles, OS9 makes frequent use of disk access to load modules into memory Those accustomed to loading an R.S. Dos programme from disk and then running or executing it, will no doubt notice the difference under OS9. A single disk drive very quickly becomes painful, especially if the format is 35 track single sided, so it is well worth the effort required to make use of that 40 track double sided drive or drives.

```
* Patches Level I Ver. 2
* CCDisk to read, write
* and format both single
* and double sided disks
t
tmode .1 -pause
save /d0/ccdisk ccdisk
debug
lccdisk
. .+7
=80
$load /d0/ccdisk
lccdisk
. .+3
=82
lccdisk
. .+1c9
=16
=01
=84
lccdisk
. .+1f8
=5f
=17
=01
=76
lccdisk
. .+2ae
=e6
=c9
=00
=a9
=16
=00
=8c
=12
=23
=02
lccdisk
. .+2dd
=40
=12
=12
=12
=17
=00
=90
. .+3
=5f
=16
=00
=81
lccdisk
. .+341
=a6
=07
=85
=01
=26
=02
=ca
=40
=a6
=09
=81
=15
=16
=ff
=66
=a6
=88
=10
=85
=01
=27
=0e
=64
=e4
=24
=0a
=c9
=00
=a9
=35
=02
=81
=15
=16
=fe
=61
=ea
=a8
=22
=16
=ff
=01
=cb
=10
=ea
=a8
=22
=34
=02
=17
=fe
=f7
=35
=02
=39
q
del /d0/ccdisk
save /d0/temp ccdisk
verify u </d0/temp >/d0/ccdisk
del temp
tmode .1 pause
-t
```

The above patch will work on Level 1 version 02.00.00 as supplied for the CoCo and will allow CCDisk to read, write and format a double sided disk.

The above is a shellscript, or procedure file if you like, which should be typed in as one long file. Just work down the left column to the bottom and then start at the top of the second, then the right column. The "build" command could be used to do this, and of course it will need a filename, CCDisk.patch or whatever.

I suggest that a backup of your 35 track boot disk should be used to perfom this patch. Once the file is created in the root directory of the backup, simply type CCDisk.patch at the OS9 prompt, and you should be left with a patched CCDisk in the root directory. A new boot disk will then have to be made using the patched version in place of the original CCDisk. The new boot disk will have to be a 35 track single sided disk.

However, there is one more step needed. The device descriptors D0, D1 etc., will also need a patch, so that OS9 knows they are for double sided drives. At an offset of $19, change the value from 1 to 2, and of course, use these patched descriptors in your new boot.

```
t
tmode .1 -pause
save /d0/d1 d1
debug
ld1
. .+19
c 1 2
q
del /d0/d1
save /d0/temp d1
verify u </d0/temp >/d0/d1
del temp
tmode .1 pause
-t
```

Repeat for d0 after changing all the occurrences of 'd1' to 'd0'

The above patches will allow the use of double sided 35 track drives giving a total of 70 tracks or 1260 sectors. However, the one catch is that the boot disk must still be a single sided disk, as the R.S.Dos boot looks for the OS9Boot file on track 34 of a single sided drive.

What happened to the 40 track double sided format you ask? The patched CCDisk now reads the track count, number of sides and head step rate from each device descriptor so we can modify the disk descriptors to what we want.

I modified D0 with the dmode utility and here is the result of a 'cmp' (compare) of the original D0 (#1) and one modified for double sides and 40 cylinders.

```
 Differences
byte      #1 #2          The procedure file above patched only one byte at offset 19 hex from 01 to 02
========  == ==          To make the drive descriptor 40 track double sided, follow the example above and change
00000018  23 28          the values at 18, 2C, 2D, 2E to those in column #2.
00000019  01 02
0000002C  ED 9C
0000002D  88 EF
0000002E  9E C4
Bytes compared:   0000002F
Bytes different:  00000005
```

Well I hope that the CCDisk patch will help some Level 1 users. As always, there is a gotcha! Because both OS9Gen and Cobbler are hard coded for 630 sectors, it is not possible (without patching them) to make a successful 40 track boot disk. I have, however, made a new 35 track, single sided boot disk using the above information, and everthing seemed to work fine. I hope to cover some further aspects of this in future newsletters.

Until next month, Regards, Gordon.

A Database in C
by Bob Devries

Here now is the save() function for my C database. This function is passed the record number, and a boolean variable to indicate whether to add the record to the end of the file or to replace the current pointed to record.

You may be interested to know that I have been able to compile this programme on my Commodore Amiga with Lattice C, but quite a few modifications had to be made. Mostly it was because the Lattice compiler is much more fussy about passing variables to functions, and how pointers are passed. It is also a compiler that adheres to the ANSI standard for C compilers, which the OS9 C compiler does not. And, of course I had to write a new 'ansi.h' file to allow for the different screen addressing.

```
save(recno,replace)
int recno;
int replace;
/* save() either saves the data held in the structure in the position of */
/* the file pointer, or at the end of the file depending on whether the */
/* variable 'replace' is TRUE (current pos) or FALSE (end of file) */
{
        long pos,lof;           /* curr pos and last of file pos */
        struct address temp;    /* temp address struct */
        int temprec;
        char ch;

        temprec = 0;
        fseek(fp,0l,2);
        lof = ftell(fp);

        if (replace == TRUE)    /* replace current record */
                {
                fseek(fp,(long)(recno - 1)*sizeof(mail),0);
                fwrite(&mail,sizeof(mail),1,fp);
                }
        else
                {               /* replace deleted record or seek to eof */
                do      {
                        temprec++;
                        fseek(fp,(long)(temprec - 1)*sizeof(mail),0);
                        pos = ftell(fp);
                        if (pos >= lof)
                                {
                                lastrec += 1;
                                break;
                                }
                        fread(&temp,sizeof(temp),1,fp);
                        ch = temp.surname[0];
                        fseek(fp,(long)(temprec - 1)*sizeof(mail),0);
                        } while (ch != '\0');
                recno = (int)ftell(fp)/sizeof(mail)+1;
                fwrite(&mail,sizeof(mail),1,fp);
                }
return(recno);
}
```

NATIONAL OS9 USER GROUP APPLICATION / RENEWAL FORM
****

[ ] SUBSCRIPTION RENEWAL [ ] NEW SUBSCRIPTION

Surname : _____ First Name :_____ Title (Mr.,etc): ____

Street : _____

Suburb : _____ State :_____ Postcode : _____

Home Phone :_____ Business Phone : _____

Do you run OS9 Level 1 [__] and/or OS9 Level 2 [__] (please tick)

Type of Computer for OS9 :_____

Diskette Size (inches): ____ Number of Tracks: ____ Sides:____ Number of Drives :__

Special Interests : _____

Can you contribute articles to this Newsletter ? _____

Date : ____/____/____ Signature :_____

Amount Enclosed: $ _____ ($18-00 will cover you for 12 months)
(cheques payable to National OS9 User Group)

NOTE: All current subscriptions Expire AUGUST 31st, 1989 (Two Year subscribers AUG 1990)

- Renewals Commence SEPTEMBER 1989 (SEPT 1990)

All current subscribers should now have back-copies from SEPTEMBER 1988 if you are missing any editions please note here the missing editions.

[ ] SEP [ ] OCT [ ] NOV [ ] DEC [ ] JAN/FEB [ ] MAR [ ] APR [ ] MAY [ ] JUN [ ] JUL [ ] AUG

Please Return Completed Form to :-

NATIONAL OS9 USER GROUP
C/o Gordon BENTZEN
8 ODIN STREET,
SUNNYBANK QLD 4109